

量子コンピュータ時代のプログラミングセミナー

～ ブラックボックス最適化を活用した材料探索 ～



本日の予定

第一部

- 本セミナーのゴール
- 会社紹介
- Fixstars Amplify の紹介
- 組合せ最適化事例
- ワークショップ事前準備

第二部

- 組合せ最適化の基本
 - 数の分割ハンズオン

第三部

- ブラックボックス最適化とは
- FMQAの概要とフロー
- FMQAによる材料探索ハンズオン
 - 問題の説明
 - FM
 - FMQA
- Amplify-BBOpt
- まとめ

質問は随時 Zoom の Q&A へお願いします

本セミナーのゴール

組合せ最適化に関する次の項目を実感していただく

- 身の回りに組合せ最適化問題は多い。一方で、複雑で非線形な物理・社会現象の場合、組合せ最適化に必要な定式化が困難なものもある。
- 機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化により、複雑現象に対する最適化を実施可能。
- ブラックボックス最適化を Fixstars Amplify により実施できる。

質問は随時ZoomのQ&Aへお願いします

会社紹介

フィックスターズグループの基本情報

会社名	株式会社フィックスターズ
本社所在地	東京都港区芝浦3-1-1 msb Tamachi 田町ステーションタワーN 28階
設立	2002年8月
上場区分	東証プライム（証券コード：3687）
代表取締役社長	三木 聡

資本金	5億5,446万円
社員数（連結）	292名（2023年9月現在）
主なお客様	キオクシア株式会社 ルネサスエレクトロニクス株式会社 トヨタグループ（トヨタ自動車株式会社・ 豊田通商株式会社・株式会社デンソー） みずほ証券株式会社 キヤノン株式会社

グループ会社

Fixstars Solutions, Inc.

完全子会社
米国での営業及び開発を担当

(株) Fixstars Autonomous Technologies

株式会社ネクスティ エレクトロニクスとのJV
自動運転向けソフトウェアを開発

(株) Fixstars Amplify

完全子会社
量子コンピューティングのクラウド事業を運営

2021/10/1 設立

(株) Sider

完全子会社
開発支援SaaS「Sider」を運営

(株) Smart Opinion

連結子会社
乳がんAI画像診断支援事業を運営

オスカーテクノロジー (株)

連結子会社
ソフトウェア自動並列化サービスを提供

フィックスターズの量子技術への取り組み

次世代技術を先取りし
今ある課題の解決を目指す

2018年

NEDOのプロジェクトに採択
「イジングマシン共通ソフトウェア
基盤の研究開発」

2017年

日本で初めて
D-Wave Systems社と提携

2019年

SIPの研究開発に参画
「光・量子を活用したSociety 5.0実現化技術：
光電子情報処理」

2021年

2月：量子アニーリングクラウドサービス「**Fixstars Amplify**」提供開始
10月：株式会社 Fixstars Amplify 設立
11月：Q-STAR 量子技術による新産業創出協議会に特別会員として加入

2022年

5月：Fixstars Amplify がGurobi, IBM-Quantumをサポート
6月：東洋経済主催シンポジウム「ビジネスを劇的に変える量子コンピューティングの可能性」に登壇
7月：累計実行回数1,000万回突破

量子技術とFixstars Amplify

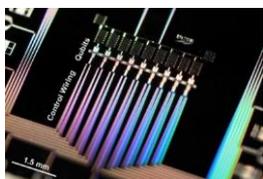
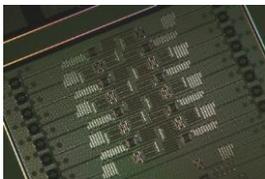
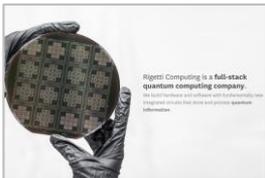
量子技術とFixstars Amplifyの対応領域

1. 量子コンピュータ

量子ゲート方式

古典汎用コンピュータの上位互換。
量子力学の重ね合わせ状態を制御する量子ゲートを操作し、特定の問題を汎用的かつ高速に処理する。

QAOAにより**組合せ最適化問題 (QUBO)**を取り扱うことが可能。



1
量子コンピュータ

IBM/Google/Rigetti/IonQ

2
量子
アニーリング
D-Wave/NEC

3
イジングマシン

富士通/日立/東芝/Fixstars

3. イジングマシン

二値二次多項式模型

二次の多変数多項式で表される目的関数の**組合せ最適化問題 (QUBO)**を扱う専用マシン。

変数は0,1または±1。統計物理学におけるイジング模型（磁性体の性質を表す模型）に由来。様々な実装により実現されている。



2. 量子アニーリング方式

量子焼きなまし法

イジングマシンの一種であり、量子焼きなまし法の原理に基づいて動作する。量子イジング模型を物理的に搭載したプロセッサで実現する。自然計算により低エネルギー状態が出力される。**組合せ最適化問題 (QUBO)**を扱う専用マシン。

組合せ最適化問題 (QUBO)

数理最適化問題

- 連続最適化問題
 - ・ 決定変数が連続値 (実数など)
- 決定変数が離散値 (整数など)
 - ・ 整数計画問題 (決定変数が整数)
 - ・ 0-1整数計画問題 (決定変数が二値)

QUBO目的関数 (0-1整数二次計画問題)

$$f(\mathbf{q}) = \sum_{i < j} Q_{ij} q_i q_j + \sum_i Q_{ii} q_i$$

f: 目的関数

q: 決定変数

Q: 係数

量子アニーリング・イジングマシン

Q uadratic	二次形
U nconstrained	制約条件なし
B inary	0-1整数 (二値)
O ptimization	計画 (最適化)



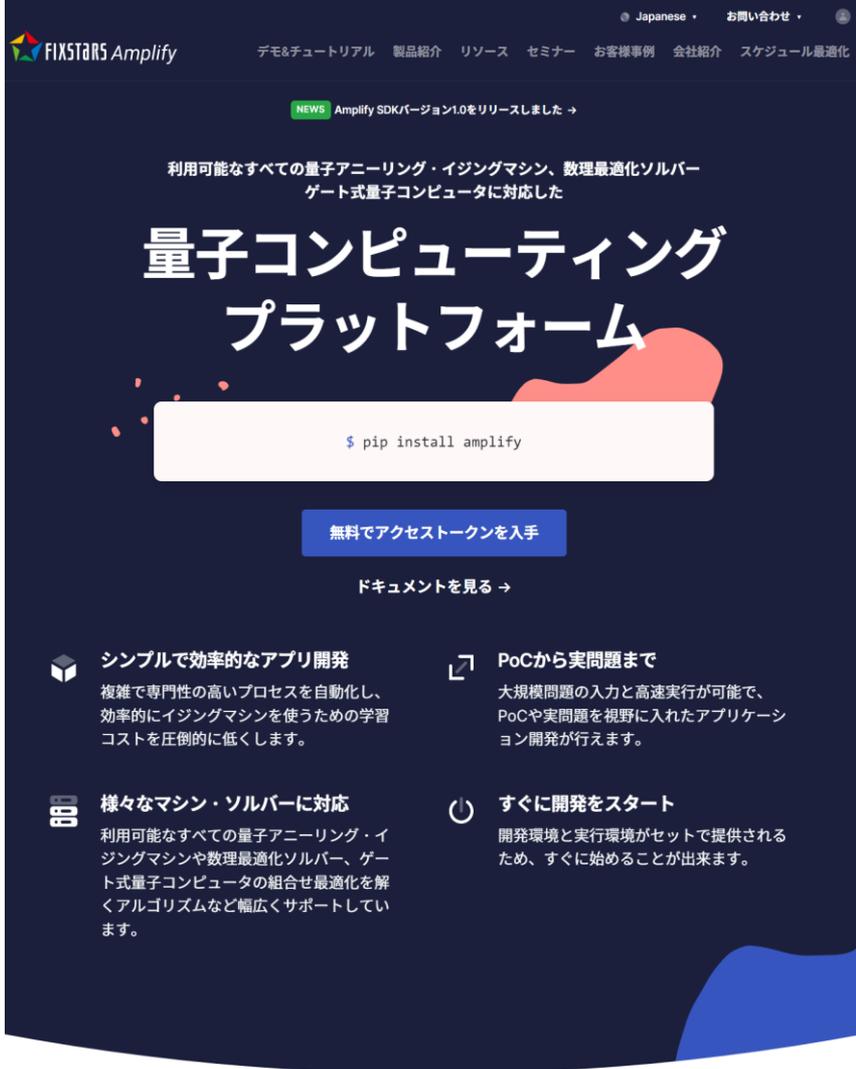
$f(\mathbf{q})$ を最小化するような \mathbf{q} を求める



クラウドサービス : **Fixstars Amplify**

Fixstars Amplify とは

- いつでも [開発環境](#) と [実行環境](#) がセット
すぐにアプリ開発と実行が出来る
- 誰でも ハードウェアや専門的な知識が不要
無料で開発がスタート可能
多くの解説、サンプルコード
- 高速に 26万ビットクラスの大規模問題の
高速処理と高速実行が可能
- あらゆる 一般に公開されている全てのイジング
マシンを利用可能



Japanese お問い合わせ

デモ&チュートリアル 製品紹介 リソース セミナー お客様事例 会社紹介 スケジュール最適化

NEWS Amplify SDKバージョン1.0をリリースしました →

利用可能なすべての量子アニーリング・イジングマシン、数理最適化ソルバー
ゲート式量子コンピュータに対応した

量子コンピューティング プラットフォーム

```
$ pip install amplify
```

無料でアクセストークン入手

ドキュメントを見る →

-  **シンプルで効率的なアプリ開発**
複雑で専門性の高いプロセスを自動化し、
効率的にイジングマシンを使うための学習
コストを圧倒的に低くします。
-  **PoCから実問題まで**
大規模問題の入力と高速実行が可能で、
PoCや実問題を視野に入れたアプリケー
ション開発が行えます。
-  **様々なマシン・ソルバーに対応**
利用可能なすべての量子アニーリング・イ
ジングマシンや数理最適化ソルバー、ゲ
ート式量子コンピュータの組合せ最適化を解
くアルゴリズムなど幅広くサポートしてい
ます。
-  **すぐに開発をスタート**
開発環境と実行環境がセットで提供される
ため、すぐに始めることができます。

Fixstars Amplify の対応マシンの一例

<p>量子アニーリング・イジングマシン</p>  <p>標準マシン</p> <p>フィックスターズ Amplify Annealing Engine</p>	<p>量子アニーリング・イジングマシン</p>  <p>標準マシン</p> <p>D-Wave Systems 2000Q / Advantage</p>	<p>量子アニーリング・イジングマシン</p>  <p>標準マシン</p> <p>東芝 デジタルソリューションズ SQBM+</p>	<p>量子アニーリング・イジングマシン</p>  <p>標準マシン</p> <p>NEC Vector Annealing</p>
<p>量子アニーリング・イジングマシン</p>  <p>富士通 デジタルアニーラ</p>	<p>量子アニーリング・イジングマシン</p>  <p>日立製作所 CMOSアニーリングマシン</p>	<p>数値最適化ソルバー</p>  <p>Gurobi Gurobi Optimizer</p>	<p>ゲート式量子コンピュータ</p>  <p>IBM IBM Quantum</p>
<p>量子回路シミュレータ</p>  <p>Qulacs Qulacs</p>	<p>様々なソルバーも 順次追加予定！</p>		

標準マシン は、

- ベンダ各社と個別マシン利用契約なし、
 - 評価・検証用ベーシックプランなら無料、
- で利用可能！ ←「いつでも」、「誰でも」

今後も幅広い対応マシンの追加が続々と行われる予定です！ ←「あらゆる」

Fixstars Amplify の内容と特徴

- 開発環境 : Amplify SDK
- 実行環境 : Amplify Annealing Engine (AE)

開発環境 : Fixstars Amplify SDK

Fixstars Amplify SDK なら最適化アプリ開発のプログラミングが圧倒的に短縮されます

通常のプログラミング

1. 課題を定式化

マシンのSDKやAPI仕様に合わせて物理モデルをデータ化

2. 論理モデルへ変換

目的関数をマシンの動作モデルで再定義

3. 物理モデルへ変換

マシン仕様や制約を考慮した物理モデルに再変換

4. マシンにデータを入力

マシンのSDKやAPI仕様に合わせて物理モデルをデータ化

5. マシンを実行

特定マシンのみで実行可能

Fixstars Amplifyを用いたプログラミング

1. 課題を定式化

定式化された数式をプログラムコードで表現

SDKが提供するAPIが、自動で各マシンに合った形式へ多段変換をして入力。実行結果は逆変換をして、ユーザーにとって結果の解釈が容易な形式で返却されます。

2. マシンを実行

複数マシンの中から選択可能

開発環境インストール

```
$ pip install amplify
```

最適化コード例

```
1 from amplify import VariableGenerator, FixstarsClient, solve
2
3 # 入力モデルの構築
4 q = VariableGenerator().array("Binary", 2)
5 f = 1 - q[0] * q[1]
6
7 # 実行マシンの設定
8 client = FixstarsClient()
9 client.parameters.timeout = 1000
10
11 # アニーリングの実行
12 result = solve(f, client)
13
14 # 結果の解釈
15 solution = q.evaluate(result.best.values)
16
17 print(f"result: {q} = {solution}")
18 # result: [q_0, q_1] = [1. 1.]
```

実行環境 : Fixstars Amplify Annealing Engine (AE)

NVIDIA GPU V100/A100 で動作

- 独自の並列化シミュレーテッド
アニーリングアルゴリズム

WEB経由で計算機能を提供

- 社会実装・PoC・検証が加速
- Amplify SDK の実装を直ぐに実行可能

商用マシンでは最大規模・最高速レベル

- 120,000 ビット (全結合)
- 260,000 ビット超 (疎結合)

	標準マシン Fixstars Amplify AE	標準マシン D-Wave 2000Q/Advantage	標準マシン 東芝 SQBM+	日立 CMOS Annealing	富士通 Digital Annealer
装置型式	GPU	量子回路	GPU	デジタル回路	デジタル回路
最大ビット数	<u>262,144以上</u>	2,048 (16x16x8)/ 5,760 (16x15x24)	100,000 (SQBM+)/ 10,000 (SBM PoC 版)	61,952 (352x176)	8,192 (DA2)/ 100,000 (DA3)
係数パラメータ	デジタル (32/64bit)	アナログ (5bit程度)	デジタル (32bit)	デジタル (3bit)	デジタル (16/64 bit)
結合グラフ	全結合	キメラグラフ/ ペガサスグラフ	全結合	キンググラフ	全結合
全結合換算ビット 数	131,072	64/124	31,000程度 (SQBM+) ^(*) / 1,000 (SBM PoC 版)	176	8,192 (DA2)/ 100,000 (DA3)
APIエンドポイン ト	Fixstars Amplify	D-Wave Leap	Fixstars Amplify / AWS	Annealing Cloud Web	DA Cloud

オンラインデモ & チュートリアル

Amplify デモ

検索

<https://amplify.fixstars.com/ja/demo>



デモアプリケーション

ピクロスパズルの求解

プログラミング難易度 ★★★★★
複雑な定式化の例として、数字で与えられるヒントを元にマスを塗り、絵を完成させるパズルゲーム、ピクロスを解くアプリケーションを開発します。

デモアプリ

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (1)

プログラミング難易度 ★★★★★
複雑で未知な目的関数にも適用可能な、機械学習と組み合わせ最適化を組み合わせたブラックボックス最適化手法を紹介し、Amplifyを用いて実装します。

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★★
機械学習と量子アニーリング・イジングマンを適用するブラックボックス最適化の適用例として、発熱炉の高温度制御を実現する材料探索を取り扱います。

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★★
化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と流体シミュレーションを用い、真の最適化と化学反応に関する物理シミュレーションを行います。

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★★
流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、最小化させ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード



デモアプリケーション

容量制約つき運搬経路問題 (CVRP)

プログラミング難易度 ★★★★★
運送業における効率的な配達計画の策定やごみ収集や道路清掃における巡回順序の最適化等での応用が期待される容量制約つき運搬経路問題 (CVRP) を取り扱います。

デモアプリ

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★★
ブラックボックス最適化により、高度最適化による交通集中が発生し得る都市における、第一別と変化する交通状況に反応し、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。また、その様な信号制御を実施した際の都市の交通量をシミュレーションします。

サンプルコード



チュートリアル応用編

定式化による交通信号機の最適化

プログラミング難易度 ★★★★★
都市における渋滞を最小化するために、第一別と変化する交通状況に反応し、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。また、その様な信号制御を実施した際の都市の交通量をシミュレーションします。

サンプルコード



チュートリアル応用編

10. 整数長ジョブスケジューリング問題

プログラミング難易度 ★★★★★
あらかじめ決まった数のジョブとマシンがあり、それぞれのジョブにかかる時間が分かっているとする。それぞれのジョブをいずれのマシンに割り当てる。ジョブスケジューリング問題では、最も早く全ジョブが完了するような割り当て方を求めます。

サンプルコード



チュートリアル基礎編

画像のノイズ除去

プログラミング難易度 ★★★★★
画像のノイズ除去を行うアプリケーションを開発します。

サンプルコード



チュートリアル応用編

会議室割当問題

プログラミング難易度 ★★★★★
制約条件を用いて定式化するアプリケーションの例として会議室割当問題のアプリケーションを開発します。

サンプルコード



チュートリアル応用編

タクシーマッチング問題

プログラミング難易度 ★★★★★
目的関数と制約条件を用いて定式化するアプリケーションの例としてタクシーマッチング問題のアプリケーションを開発します。

サンプルコード



デモアプリケーション

グラフ彩色問題

プログラミング難易度 ★★★★★
Fixstars Amplifyによる、グラフ彩色問題の定式化を体験します。

デモアプリ

サンプルコード



デモアプリケーション

巡回セールスマン問題

プログラミング難易度 ★★★★★
Fixstars Amplifyによる、巡回セールスマン問題の定式化を体験します。

デモアプリ

サンプルコード



デモアプリケーション

数独

プログラミング難易度 ★★★★★
Fixstars Amplifyによる、数独の定式化を体験します。

デモアプリ

サンプルコード



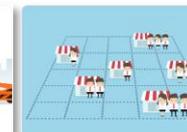
デモアプリケーション

ライドシェア

プログラミング難易度 ★★★★★
集合型ライドシェアの最適化アプリケーションを体験します。

デモアプリ

サンプルコード



デモアプリケーション

タスク割当問題

プログラミング難易度 ★★★★★
店舗とタスクに従業員を割り当てる組合せ最適化問題のアプリケーションを体験します。

デモアプリ

サンプルコード



デモアプリケーション

ポートフォリオ最適化

プログラミング難易度 ★★★★★
リスクとリターンを考慮した株式ポートフォリオの最適化アプリケーションを体験します。

デモアプリ

サンプルコード

様々な領域での利用拡大中（実稼働含む）

900を超える企業、研究所、大学

8,000万を超える実行回数 (Amplify AE)



Fixstars Amplify ご利用プラン

料金のご紹介

<https://amplify.fixstars.com/ja/pricing>

	ベーシック 評価・検証用の無料プラン	スタンダード 実用レベルの計算環境	プレミアム 高性能な計算環境	Sプレミアム 最高性能の計算環境
	使い始める	見積りを依頼	見積りを依頼	見積りを依頼
利用料金	無料	月額10万円 (1名) (税込11万円) 月額20万円 (最大5名) (税込22万円)	月額20万円 (1名) (税込22万円) 月額60万円 (最大5名) (税込66万円)	月額30万円 (1名) (税込33万円) 月額90万円 (最大5名) (税込99万円)
計算環境	スモール	ミディアム	ラージ	スーパーラージ
D-Waveマシンの無料実行	3分/月	3分/月	3分/月	3分/月
SQBM+オプション	無料	月額30万円 (1名) (税込33万円) 月額90万円 (最大5名) (税込99万円)	月額30万円 (1名) (税込33万円) 月額90万円 (最大5名) (税込99万円)	月額30万円 (1名) (税込33万円) 月額90万円 (最大5名) (税込99万円)
サポート	ベーシック	スタンダード	プレミアム	プレミアム
評価・検証フェーズでの利用				
実運用フェーズでの利用				

開発支援サービス(個別見積り)

コンサル・システム開発等
数百万円～数千万円



月額利用料
百万円～

定式化や実装を
手厚く
支援します！

セミナー・トレーニングのご紹介

<https://amplify.fixstars.com/ja/news/seminar>

お客様の実際の課題解決をご支援するために、**無料セミナー**や**有償トレーニング**を提供しています。

無料セミナー・ワークショップ

ビジネス向け、エンジニア向けに分けて開催しています！

ビジネス向け

製造業向け量子コンピュータ時代のDXセミナー 見える化、予測・分析、その先の最適化へ

組合せ最適化問題や量子アニーリング・イジングマシンの概要をご紹介したのち、製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化や生産ラインのシフト最適化などの事例とデモをご紹介します。「Fixstars Amplify」を通じて量子アニーリング・イジングマシンを活用することで、どのようなビジネス上の効果が期待できるのかを感じていただきたいと思います。

エンジニア向け

製造業向け量子コンピュータ時代のDXセミナー 最適化の中身を覗いてみよう

製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化、勤務シフト最適化などの事例を用いて、問題設定の考え方、目的関数や制約条件の定式化、実装のポイントなどを実際のコードを見ながら解説します。また、サンプルコードを用いて、ご自身の環境で実際に量子アニーリング・イジングマシンを動かす体験をしていただけます。

企業向けプライベートトレーニング

お客様が抱える実際の課題やデータを使った**カスタムメイド**のトレーニングです！

全4回のレクチャーとお客様に実施いただく「課題」を含む約1.5か月のコースです。コースの前半では、量子アニーリング・イジングマシン専用の開発／実行環境であるFixstars Amplifyを用いてPython言語による組合せ最適化アプリケーション開発方法を学びます。後半では、お客様が抱える実際の課題やデータを使ったトレーニングを実施します。量子アニーリング・イジングマシンを使って実課題の解決に取り組んでみたい方に最適なコースです。

第1回
3時間

…
1週間

第2回
3時間

課題
2週間

第3回
1.5時間

…
2週間

第4回
1.5時間

ワークショップ°

事前準備（事前メールの内容）

ワークショップの事前準備 (1)

- 【事前メールに記載】ご自身のPC (ブラウザ上) で Python プログラミングを行います。Google Colaboratory を使うので、事前にログイン出来ることを確認をお願いします (要 Google アカウント)

Google Colab 検索

<https://colab.research.google.com/>

- 【事前メールに記載】 Fixstars Amplify ホームページよりユーザ登録の上、無料トークンの取得をお願いします (1分で終わります)

Fixstars Amplify 検索

<https://amplify.fixstars.com/>



質問は随時ZoomのQ&Aへお願いします



ワークショップの事前準備 (2)

【事前メールに記載】

- 取得されたトークンを用いて、トークンチェック用サンプルコードが動くか確認をお願いします。

https://colab.research.google.com/drive/1bg2Ql3McJck_Sto8uvxtmPUMWtRFhf7a (※URLはZoomのチャット欄を参照)

- サンプルコードは閲覧のみ可能な状態です。「ファイル」→「ドライブにコピーを保存」の上、ご自身のトークンを入力してください。その後、Shift + Enterで実行下さい。

```
! pip install amplify
```

```
token = "AE/*****" # ご自身のトークンを入力
```

- ご自身のトークン番号は、Amplifyウェブページ → よりご確認いただけます。
- 実行後、以下の結果が出力されればOKです。

```
result: [q_0, q_1] = [1. 1.] (f = 0.0)
```



ワークショップ^o

通常の組合せ最適化

(ブラックボックス最適化への導入)

サンプルコード

サンプルコードを開き、「ファイル」→「ドライブにコピー」の上、**トークンを入力**し実行して下さい。

- 数の分割サンプルコード

<https://colab.research.google.com/drive/1bl6BGPwnpd3N0AveoDMGb8egprEdQMqU>

(※URLはZoomのチャット欄を参照)

質問は随時 Zoom の Q&A へお願いします

数の分割問題（概要）

- 与えられた n 個の整数 a_0, \dots, a_{n-1} を二つの集合に分ける。
集合内の数の和が、もう一方の集合内の数の和と等しくなることができるか？
 - NP完全問題: とても難しい問題として知られている → 全通り試すしか方法は無い
 - 問題のバリエーション
 - 判定問題: 完全に等しく出来るか？または等しい組合せは何か？
 - 最適化問題: 完全に等しいか、または最も惜しい組合せは何か？



数の分割問題（具体例と解法の方針）

具体例

{2, 10, 3, 8, 5, 7, 9, 5, 3, 2} の10個の数の完璧な分割は見つけられるか？

答え

- 存在する
 - {2, 3, 5, 7, 10} と {2, 3, 5, 8, 9}
 - どちらも和は 27
- 分割方法は 23 通り存在する (対称を除く)

どうやって解くか？

- ひとつの『数』がどちらの集合に分割されるか全通り試す → $2^{10} = 1024$ 通り

- 効率のよい厳密な方法は知られていない・・・ (もし発見されたら大騒ぎ)

数の分割問題（定式化）

最適化問題：数の分割において最も惜しい組合せは何か？

- 目的関数

{集合1の和} - {集合2の和} の絶対値を最小化

- 決定変数

数 a_i がどちらの集合に属するかを s_i で表す

- $a_i = \{ 2, 10, 3, 8, 5, 7, 9, 5, 3, 2 \}$

- $s_i = \{-1, 1, -1, 1, -1, -1, 1, 1, 1, 1\}$

数理モデル

- 目的関数

$$f = \left| \sum_{i=0}^{N-1} s_i a_i \right| \quad (s_i \in \{-1, +1\})$$

$\sum s_i a_i$ は、自然と
{『1』の集合の和} - {『-1』の集合の和}
となる！

数の分割問題（バイナリへの式変形）

- 0-1整数二次計画問題への変換
 - Quadratic Unconstrained Binary Optimization (QUBO) 式

$$f = \left| \sum_{i=0}^{N-1} s_i a_i \right| \quad (s_i \in \{-1, +1\})$$

$\sum s_i a_i$ は、自然と
{『1』の集合の和} - {『-1』の集合の和}
となる！

$$\rightarrow \left(\sum_{i=0}^{N-1} s_i a_i \right)^2 \quad (s_i \in \{-1, +1\})$$

絶対値を二次式で表す

$$\rightarrow \left(\sum_{i=0}^{N-1} (2q_i - 1) a_i \right)^2 \quad (q_i \in \{0, +1\})$$

± 1 をバイナリで表す

数の分割問題（定式化の具体例）

問題

- $a_i = \{2, 10, 3, 8, 5, 7, 9, 5, 3, 2\}$ の10個の数の完璧な分割は見つけれられるか？

決定変数

- $q_i = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\} (q_i \in \{0, 1\})$ で集合0又は集合1、どちらに所属するかを表す

目的関数

$$f = \left(\sum_{i=1}^N (2q_i - 1)a_i \right)^2$$

目的関数を展開

$$f = \left(\begin{array}{l} 2(2q_0 - 1) + 10(2q_1 - 1) + 3(2q_2 - 1) + 8(2q_3 - 1) + 5(2q_4 - 1) \\ + 7(2q_5 - 1) + 9(2q_6 - 1) + 5(2q_7 - 1) + 3(2q_8 - 1) + 2(2q_9 - 1) \end{array} \right)^2$$

数の分割問題（プログラムコード）

- 問題の定義と決定変数生成器による決定変数の生成

```
a = [2, 10, 3, 8, 5, 7, 9, 5, 3, 2]
q = amplify.VariableGenerator().array("Binary", len(a))
```

1. 定式化

- 目的関数、 $f = (\sum_{i=1}^N (2q_i - 1)a_i)^2$ 、の定式化（①②③は同等）

① `f = ((2 * q - 1) * a).sum() ** 2`

② `f = 0`
`for i in range(len(a)):`
`f += (2 * q[i] - 1) * a[i]`
`f **= 2`

③ `f = amplify.sum((2 * q - 1) * a) ** 2`

色々な書き方が出来る

2. 実行

```
result = amplify.solve(f, client)
```

得られた目的関数の値0

各集合の合計値27

3. 結果

```
q = [1, 1, 1, 0, 1, 1, 0, 0, 0, 0], f = 0.0, w = 27
```

各数字に対して、集合0か、集合1か

オンラインデモ & チュートリアル

Amplify デモ

検索

<https://amplify.fixstars.com/ja/demo>



デモアプリケーション

ピクロスパズルの求解

プログラミング難易度 ★★★★★
複雑な定式化の例として、数字で与えられるヒントを元にマスを塗り、絵を完成させるパズルゲーム、ピクロスを解くアプリケーションを開発します。

デモアプリ サンプルコード



チュートリアル応用編

ブラックボックス最適化 (1)

プログラミング難易度 ★★★★★
複雑で未知な目的関数にも適用可能な、機械学習と組み合わせ最適化を組み合わせたブラックボックス最適化手法を紹介し、Amplifyを用いて実装します。

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★★
機械学習と量子アニーリング・イジングマンを適用するブラックボックス最適化の適用例として、発光性高温超伝導を実現する材料探索を取り扱います。

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★★
化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と流体シミュレーションを用い、真の最適化を最大化するように異型の探索を行います。

サンプルコード



チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★★
流体機器設計に不可欠な異型の最適化問題を取り上げます。最適化には、組み合わせ最適化と機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、真の最適化を最大化するように異型の探索を行います。

サンプルコード



デモアプリケーション

容量制約つき運搬経路問題 (CVRP)

プログラミング難易度 ★★★★★
運送業における効率的な配達計画の策定やごみ収集や道路清掃における巡回順序の最適化等での応用が期待される容量制約つき運搬経路問題 (CVRP) を取り扱います。

デモアプリ サンプルコード



チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★★
ブラックボックス最適化により、高度最適化による交通集中が発生し得る都市における、第一別と変化する交通状況に応じ、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード



チュートリアル応用編

定式化による交通信号機の最適化

プログラミング難易度 ★★★★★
都市における渋滞を最小化するために、第一別と変化する交通状況に応じ、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。また、その様な信号機制御を実施した際の都市の交通量をシミュレーションします。

サンプルコード



チュートリアル応用編

10. 整数長ジョブスケジューリング問題

プログラミング難易度 ★★★★★
あらかじめ決まった数のジョブとマシンがあり、それぞれのジョブにかかる時間が分かっているとする。それぞれのジョブをいずれのマシンに割り当てます。ジョブスケジューリング問題では、最も早く全ジョブが完了するような割り当て方を求めます。

サンプルコード



チュートリアル基礎編

画像のノイズ除去

プログラミング難易度 ★★★★★
画像のノイズ除去を行うアプリケーションを開発します。

サンプルコード



チュートリアル応用編

会議室割当問題

プログラミング難易度 ★★★★★
制約条件を用いて定式化するアプリケーションの例として会議室割当問題のアプリケーションを開発します。

サンプルコード



チュートリアル応用編

タクシーマッチング問題

プログラミング難易度 ★★★★★
目的関数と制約条件を用いて定式化するアプリケーションの例としてタクシーマッチング問題のアプリケーションを開発します。

サンプルコード



デモアプリケーション

グラフ彩色問題

Fixstars Amplifyによる、グラフ彩色問題の定式化を体験します。

デモアプリ サンプルコード



デモアプリケーション

巡回セールスマン問題

Fixstars Amplifyによる、巡回セールスマン問題の定式化を体験します。

デモアプリ サンプルコード



デモアプリケーション

数独

Fixstars Amplifyによる、数独の定式化を体験します。

デモアプリ サンプルコード



デモアプリケーション

ライドシェア

集合型ライドシェアの最適化アプリケーションを体験します。

デモアプリ サンプルコード



デモアプリケーション

タスク割当問題

店舗とタスクに従業員を割り当てる組合せ最適化問題のアプリケーションを体験します。

デモアプリ サンプルコード



デモアプリケーション

ポートフォリオ最適化

リスクとリターンを考慮した株式ポートフォリオの最適化アプリケーションを体験します。

デモアプリ サンプルコード

組合せ最適化と ブラックボックス最適化

組合せ最適化とブラックボックス最適化

通常の場合の最適化

- 課題に対し、2次定式化 (QUBO) を実施

- 数の分割

$$f = [\sum(2q_i - 1)a_i]^2$$

- 生産計画最適化

$$f = \sum \sum (\sum a_i p_i q_{m,i,t} + \sum \sum b_i s_i q_{m,i,t} q_{m,i-1,f}) \dots$$

- TSP問題 (お遍路巡り最短経路探索)

$$f = \sum \sum \sum d_{i,j} q_{n,i} q_{n+1,j} \dots$$

- QUBO式に対し、直接アニーリングを行う
(量子アニーリング・イジングマシン)

ブラックボックス最適化

- 課題が複雑で直接の定式化が不可能。

例) 最も計測精度が高くなるセンサー群の配置は？

→ 実験かシミュレーション (コスト大)

- シミュレーションや実験の試行回数の削減

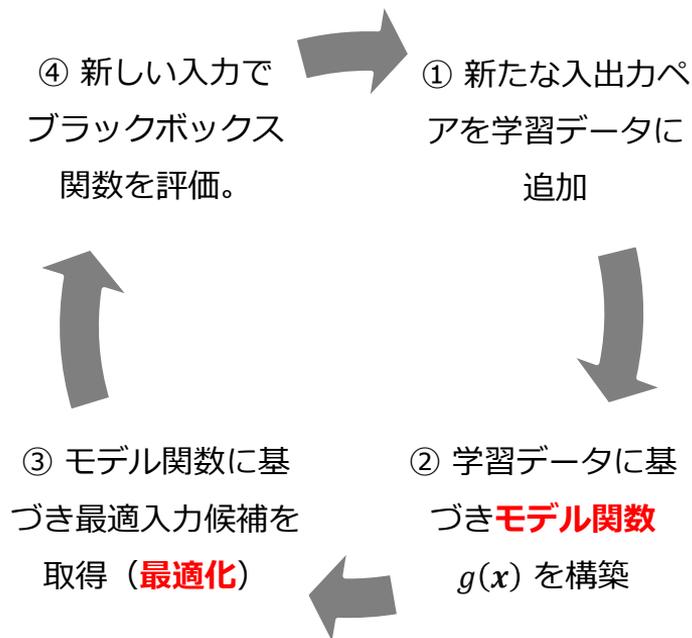
→ シミュレーションや実験が楽になる

- 逆問題に対するアプローチとしても適用化

例) ある計測値を実現する実験条件を見つける

ブラックボックス最適化アプローチ

- 次の最適化サイクルを **ブラックボックス関数 $f(x)$** を対象に実施
(実験計測や数値シミュレーション)



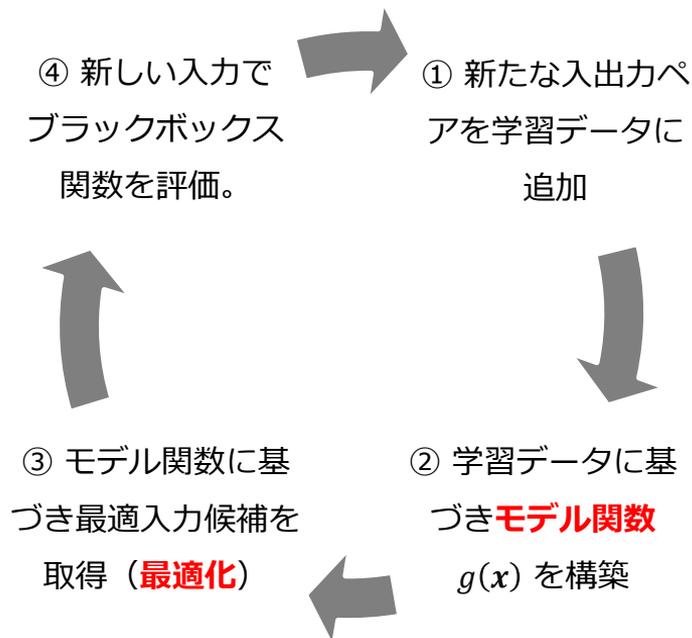
課題：

- モデル関数 $g(x)$** の構築手法
- モデル関数の**最適化**手法
※ $g(x)$ 最小を実現する x の取得

より少ないサイクル（評価回数）で
より良い解が得られるよう工夫

ブラックボックス最適化アプローチ

- 次の最適化サイクルを **ブラックボックス関数 $f(x)$** を対象に実施
(実験計測や数値シミュレーション)



FMQA : [K. Kitai, et al., Phys. Rev. Res. (2020).
T. Inoue, et al., Opt. Express (2022).]

- **モデル関数 $g(x)$**
Factorization Machine
(FM、機械学習モデルの一種)
- **最適化**
学習モデルをQUBOとして
Amplify で求解

モデル関数としての Factorization Machine (FM)

- モデル関数 $g(\mathbf{x})$ に機械学習モデルの一種である Factorization Machine (FM) を用いると、次のように変数 \mathbf{x} に対する2次式での記述ができる。

$$g(\mathbf{x}|\mathbf{w}, \mathbf{v}) = w_0 + \langle \mathbf{w}, \mathbf{x} \rangle + \sum_{i=1}^n \sum_{j=i+1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$
$$= w_0 + \sum_{i=1}^n w_i x_i + \frac{1}{2} \sum_{f=1}^k \left(\left(\sum_{i=1}^n v_{if} x_i \right)^2 - \sum_{i=1}^n v_{if}^2 x_i^2 \right) \quad \rightarrow \text{QUBO式}$$

- k はハイパーパラメータ、 \mathbf{w} 及び \mathbf{v} は FM 学習後に取得される FM パラメータ。
- FM パラメータ数は k に依存。 $k = n$ のときは QUBO の相互作用項と同じ自由度がある一方、 k を小さくすることでパラメータ数を減らし過学習を抑制する効果
- このようなブラックボックス最適化手法を **FMQA** と呼ぶ。

ブラックボックス最適化 活用例

材料分野に限らず、幅広い分野へ適用可能

QA-BBO: 活用例 (Amplify サンプルプログラム)

Amplify デモ

検索



チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★★

機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超電導を実現する材料探索を取り扱います。

サンプルコード

材料最適化

FMQA

×

物理モデル



チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★★

化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と化学反応に関する物理シミュレーションを用います。

サンプルコード

化学プラント 運転条件最適化

FMQA

×

化学シミュレーション



チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★★

流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード

翼形状最適化

FMQA

×

流体シミュレーション



チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★★

ブラックボックス最適化により、商業施設による交通集中が発生し得る都市における、交通渋滞を低減するような信号機群の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード

信号制御最適化

FMQA

×

マルチ・エージェント・シミュレーション



チュートリアル応用編

ブラックボックス最適化 (6)

プログラミング難易度 ★★★★★

ブラックボックス最適化により、攪拌性能に影響を与える設計パラメータに対して、混合効率が最大化されるような攪拌機の最適設計を実施します。最適化の実施および評価には、濃度分布に基づく簡易的な攪拌シミュレーションを用います。

サンプルコード

機器設計最適化

FMQA

×

攪拌シミュレーション

QA-BBO: 活用例 (Amplify ユーザー)

• 活用領域

- 化学、創薬、食品、自動車、電機、通信、重工、エネルギー、ヘルスケア・・・

非線形現象の逆
問題

機械学習：
コスト↓精度↑

設計開発におけ
る部品選定

材料配合最適化

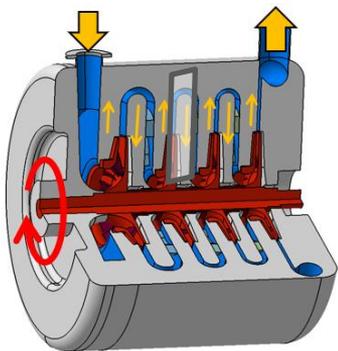
多目的最適化

物理モデルの
簡略化

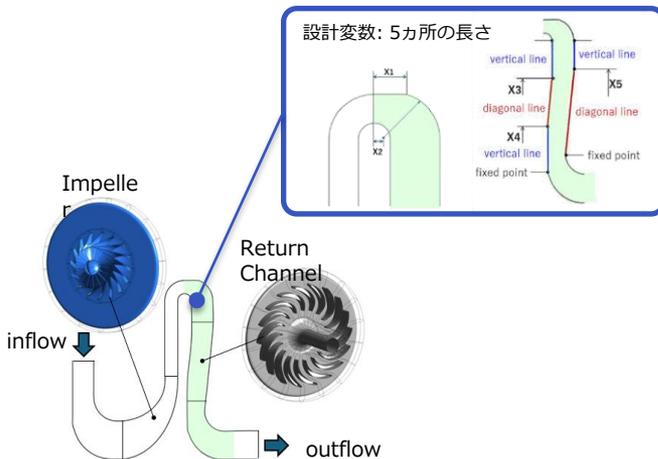
事例: ターボ機械の形状最適化 (川崎重工業様)

- ターボ機械の開発では、従来より商用最適化ソフトによる遺伝的アルゴリズム (GA) を使用し形状最適化を行うことが多かったが、最適化規模が大きくなると最適解の求解までに時間がかかり、開発期間が長期化するという課題があった
- 量子アニーリング・イジングマシンを活用した BBO により、従来手法と比べ、同じ計算回数でもより優れた解が得られることを確認。今後はさらに設計変数を増やしていく予定

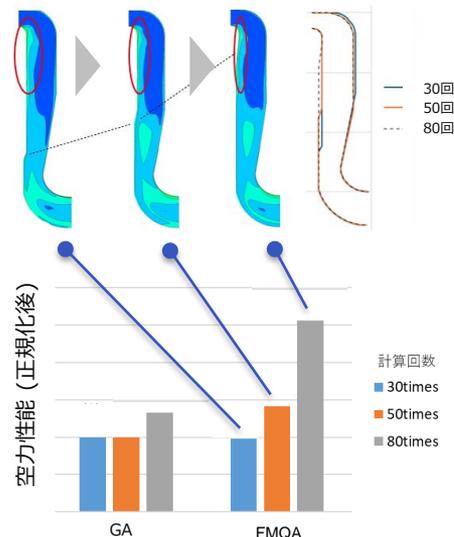
遠心圧縮機



流路形状を最適化したい
(圧縮機全体の空力性能 (ポリトロープ効率) の最大化)



最適化が進むごとに損失発生領域
が低減する形状へ



事例: 車両設計最適化 (マツダ様)

amplify マツダ

Search

- 複数車種の車体構造同時設計最適化問題。衝突性能を含めた品質特性の条件を満たした上で、部品の軽量化と共通部品数の最大化の実現する多目的最適化問題
- 2017年にマツダ・JAXA がベンチマーク問題として公開し*1、国内外の研究グループ*2,3 により様々な手法が試されていた。1~3万回程度の試行により、ある程度よい解が得られることは確認されていた

- 
- FMQA により、1,000回程度の試行で、従来手法と同等以上の解を見つけることに成功!
 - 今後は、QUBO 式近似の計算コストを削減しつつ、最適化性能も向上させるような手法の検討を予定

*1 [応答曲面法を用いた複数車種の同時最適化ベンチマーク問題の提案](#)

*2 [進化計算コンペティション2017開催報告](#)

*3 [Multi-objective Bayesian optimization over high-dimensional search spaces](#)



ブラックボックス最適化

1. 機械学習 (FM) ハンズオン
2. FMQA ハンズオン

ブラックボックス最適化のデモプログラム

模擬超電導材料の臨界温度の FM 機械学習

<https://colab.research.google.com/drive/1T74GyN07bhvfMzNltloEgBS0fW4bxtGu>

ブラックボックス関数

- ブラックボックス関数 $f(x)$ として、与えられた材料組合せによる模擬超電導材料の臨界温度を返却するモデル関数を考慮

※本関数はいくまでも実験やシミュレーションの代用であり、その中身やパラメータについては未知であるとして扱い、関数評価の回数にも制限があるものとして取り扱います。

最適化サイクル

1. 教師データから獲得関数 $g(x)$ を構築 (FM) ←
2. 獲得関数 $g(x)$ が最小となる点 \hat{x} を推定 (QA)
3. 評価結果 $(\hat{x}, f(\hat{x}))$ を教師データに追加

↑のサイクルにより、最適化点近傍におけるFMの予測精度が向上し、組合せ最適化により、より良い \hat{x} の推定が期待される

FM デモプログラム 1/5 (ブラックボックス関数の定義)

```
def make_blackbox_func(d: int) -> Callable[[np.ndarray], float]:
    """ 入力が長さ d のバリナリ値のベクトルで出力が float であるような関数を返却する """

def set_properties(size: int) -> tuple[np.ndarray, np.ndarray, np.ndarray]:
    """ (ランダムに決定した) 材料物性係数を返却する """
    mu, sigma, ratio = 0.0, 1.0, 0.2
    table1 = rng.random(size) * 1e5 * (0.1 * math.log(size) - 0.23)
    table2 = rng.lognormal(mu, sigma, size) * ratio
    table3 = rng.lognormal(mu, sigma, size) * ratio
    return table1, table2, table3

def supercon_temperature(x, debye_table, state_table, interaction_table):
    """ 与えられた材料の組合せ (長さ d のバリナリ値のベクトル) に対し、臨界温度を計算し返却する """
    debye_temperature = np.sum(x * debye_table) / np.sum(x)
    state_density = np.sum(x * state_table) / np.sum(x)
    interaction = np.sum(x * interaction_table) / np.sum(x)
    crit_temp = debye_temperature * math.exp(-1.0 / state_density / interaction)
    return crit_temp

# 係数テーブルの準備
debye_temperature_table, state_density_table, interaction_table = set_properties(d)

# ブラックボックス関数の定義
def blackbox(x: np.ndarray) -> float:
    """ 与えられた材料の組合せ (長さ d のバリナリ値のベクトル) に対し、超電導臨界温度の負値を返却する """
    assert x.shape == (d,) # x は要素数 d の一次元配列
    t_c = supercon_temperature(
        x, debye_temperature_table, state_density_table, interaction_table
    )
    return -t_c

return blackbox
```

関数の利用例 :

```
▶ num_materials = 100 # 決定変数のサイズ (材料選択肢の数)
  blackbox_func = make_blackbox_func(num_materials)
  critical_temp = blackbox_func(x)
```

FM デモプログラム 2/5 (FMの定義)

```
class TorchFM(nn.Module):
    def __init__(self, d: int, k: int):
        """モデルを構築する

        Args:
            d (int): 入力ベクトルのサイズ
            k (int): パラメータ k
        """
        super().__init__()
        self.d = d
        self.v = nn.Parameter(torch.randn((d, k)))
        self.w = nn.Parameter(torch.randn((d,)))
        self.w0 = nn.Parameter(torch.randn(()))

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        """入力 x を受け取って y の推定値を出力する

        Args:
            x (torch.Tensor): (データ数 × d) の 2 次元 tensor

        Returns:
            torch.Tensor: y の推定値 の 1次元 tensor (サイズはデータ数)
        """
        out_linear = torch.matmul(x, self.w) + self.w0
        out_1 = torch.matmul(x, self.v).pow(2).sum(1)
        out_2 = torch.matmul(x.pow(2), self.v.pow(2)).sum(1)
        out_quadratic = 0.5 * (out_1 - out_2)

        out = out_linear + out_quadratic
        return out
```

- FM モデルを PyTorch で定義する

$$f(\mathbf{x}|\mathbf{w}, \mathbf{v}) = \underbrace{w_0 + \sum_{i=1}^d w_i x_i}_{\text{out_linear}} + \frac{1}{2} \left[\underbrace{\sum_{f=1}^k \left(\sum_{i=1}^d v_{if} x_i \right)^2}_{\text{out_1}} - \underbrace{\sum_{f=1}^k \sum_{i=1}^d v_{if}^2 x_i^2}_{\text{out_2}} \right]_{\text{out_quadratic}}$$

FM デモプログラム 3/5 (FMの学習)

```
def train(
    x: np.ndarray, y: np.ndarray, model: TorchFM, epochs: int = 2000, lr: float = 1, plo

    # モデルの最適化関数
    optimizer = torch.optim.AdamW([model.v, model.w, model.w0], lr=lr)
    # 学習率スケジューラ
    scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=200, gamma=0.9)
    # 損失関数
    loss_func = nn.MSELoss()
    # データセットの用意
    x_tensor, y_tensor = (
        torch.from_numpy(x).float(),
        torch.from_numpy(y).float(),
    )
    dataset = TensorDataset(x_tensor, y_tensor)
    train_set, valid_set = random_split(dataset, [0.8, 0.2])
    batch_size = 8
    train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True)
    valid_loader = DataLoader(valid_set, batch_size=batch_size, shuffle=True)

    # 学習の実行
    min_loss = 1e18 # 損失関数の最小値を保存
    max_corrcoef = -1e18 # 相関係数の最大値を保存
    losses: list[float] = [0.0] * epochs
    corrcoefs: list[float] = [0.0] * epochs
    best_state = model.state_dict() # 最も良いモデルのパラメータを保存するための変数
    for i in trange(epochs, leave=False):
        # 学習フェイズ
        for x_train, y_train in train_loader:
            optimizer.zero_grad()
```

FM 学習は、通常の機械学習と同様に進める。教師データを学習・検証データに分割し、ミニバッチ学習。

- **x, y**: 教師データ
- **model**: FM モデル (TorchFM)
- **epochs**: エポック (繰り返し) の数
- **lr**: (初期) 学習率
- **scheduler**: 学習率スケジューラー

学習後、最も良いモデルに対し、次のような評価を実施

```
# 教師データの入力値に基づきモデルを評価
y_pred = model(x_tensor)
print(f"corrcoef: {torch.corrcoef(torch.stack([y_tensor, y_pred]))[0]
print(f"RMS error: {min_loss:.2f}")
```

FM デモプログラム 4/5 (教師データ作成)

```
def init_training_data(d: int, n0: int):
    """n0 組の初期教師データを作成する"""
    assert n0 < 2**d

    # n0 個の長さ d の入力値を乱数を用いて作成
    x = rng.choice(np.array([0, 1]), size=(n0, d))

    # 入力値の重複が発生していたらランダムに値を変更して回避する
    x = np.unique(x, axis=0)
    while x.shape[0] != n0:
        x = np.vstack((x, np.random.randint(0, 2, size=(n0 - x.shape[0], d))))
        x = np.unique(x, axis=0)

    # blackbox 関数を評価して入力値に対応する n0 個の出力を得る
    y = np.zeros(n0)
    for i in range(n0):
        y[i] = blackbox_func(x[i])

    return x, y
```

```
N0 = 10 # 初期教師データの数
x_init, y_init = init_training_data(num_materials, N0)
```

教師データを乱数により生成

- **d** : 入力サイズ
- **n0** : 教師データのサンプル数
- **blackbox** : ブラックボックス関数 (実験又はシミュレーション) $f(x)$

FM デモプログラム 5/5 (メイン部分)

```
# 乱数シードの固定
rng = np.random.default_rng(seed)

# 教師データの初期化
x, y = x_init, y_init

# 機械学習モデルの作成
model = TorchFM(d=num_materials, k=10)

# モデルの学習の実行
train(x, y, model, epochs=2000, lr=1, plot_learning_curve=True)
```

実際にサンプルプログラムを実行してみま
しょう。デフォルトの条件から、

- FMのハイパーパラメータ (k)
- エポック数 (epochs)
- 学習率 (lr)

などを変更した場合、真値と予測値の相関
係数及びRMS誤差はどのように変化するで
しょうか？

ブラックボックス最適化

1. 機械学習 (FM) ハンズオン
2. FMQA ハンズオン

ブラックボックス最適化のデモプログラム

ブラックボックス最適化による模擬超電導材料の探索

https://colab.research.google.com/drive/1E3Cy6Fe4EG-M11E_33-Djp93aSZAUiQz

ブラックボックス関数

- ブラックボックス関数 $f(x)$ として、与えられた材料組合せによる模擬超電導材料の臨界温度を返却するモデル関数を考慮

※本関数はいくまでも実験やシミュレーションの代用であり、その中身やパラメータについては未知であるとして扱い、関数評価の回数にも制限があるものとして取り扱います。

最適化サイクル

1. 教師データから獲得関数 $g(x)$ を構築 (FM)
2. 獲得関数 $g(x)$ が最小となる点 \hat{x} を推定 (QA)
3. 評価結果 $(\hat{x}, f(\hat{x}))$ を教師データに追加

↑のサイクルにより、最適化点近傍におけるFMの予測精度が向上し、組合せ最適化により、より良い \hat{x} の推定が期待される

FMQA デモプログラム 1/2 (アニーリング部分)

```
def anneal(torch_model: TorchFM) -> np.ndarray:
    """FM モデルを受け取り、それらのパラメータにより記述されるモデルの最小値を与える x を求める"""

    x = VariableGenerator().array("Binary", torch_model.d) # 長さ d のバイナリ変数の配列を作成

    v, w, w0 = torch_model.get_parameters() # TorchFM からパラメータ v, w, w0 を取得

    # 目的関数を作成
    out_linear = w0 + (x * w).sum()
    out_1 = ((x[:, np.newaxis] * v).sum(axis=0) ** 2).sum() # type: ignore
    out_2 = ((x[:, np.newaxis] * v) ** 2).sum()
    objective: Poly = out_linear + (out_1 - out_2) / 2

    amplify_model = Model(objective) # 組合せ最適化モデルを構築

    # ソルバーの設定
    client = FixstarsClient()
    client.token = token
    client.parameters.timeout = timedelta(milliseconds=2000)

    # 最小化を実行
    result = solve(amplify_model, client)
    if len(result.solutions) == 0:
        raise RuntimeError("No solution was found.")

    # モデルを最小化する入力ベクトルを返却
    return x.evaluate(result.best.values).astype(int)
```

学習済みFMに基づき \hat{x} を推定する関数

- 決定変数配列の作成
- 学習済みモデルからパラメータを取得
- パラメータに基づき目的関数 $g(x)$ を作成
- ソルバーの設定
 - ※トークン入力を忘れずに
- solve の実行
- このサイクルにおける \hat{x} を返却

FMQA デモプログラム 2/2 (メイン部分)

```
N = 10 # FMQA サイクルの実行回数
x, y = x_init, y_init # 教師データの初期化

# N 回のイテレーションを実行。`range` の代わりに `tqdm` モジュールを用いて進捗を表示
for i in trange(N):
    model = TorchFM(d=num_materials, k=10) # 機械学習モデルの作成

    train(x, y, model, epochs=1000) # モデルの学習の実行

    x_hat = anneal(model) # 学習済みの最小値を与える入力ベクトルの値を取得

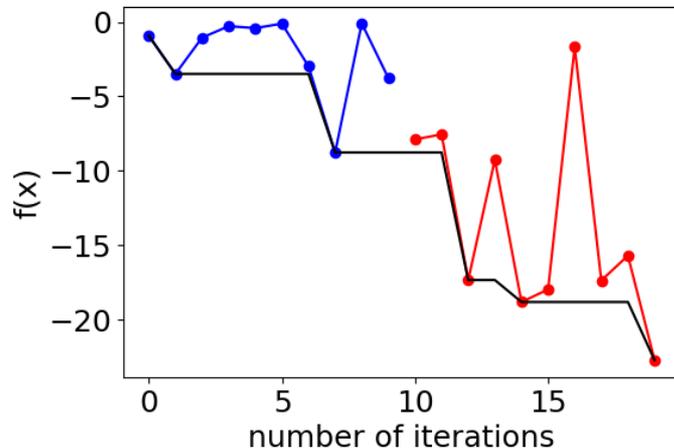
    # x_hat が重複しないようにする
    while (x_hat == x).all(axis=1).any():
        flip_idx = rng.choice(np.arange(num_materials))
        x_hat[flip_idx] = 1 - x_hat[flip_idx]

    y_hat = blackbox_func(x_hat) # x_hatを用いてブラックボックス関数を評価

    # 評価した値をデータセットに追加
    x = np.vstack((x, x_hat))
    y = np.append(y, y_hat)

    tqdm.write(f"FMQA cycle {i}: found y = {y_hat:.1f}; current best = {np.min(y):.1f}")
```

実際にサンプルプログラムを実行してみましょう！



FMQA デモプログラムの実務での活用方法

```
def make_blackbox_func(d: int) -> Callable[[np.ndarray], float]:  
    rng = np.random.default_rng(seed)  
    Q = rng.random((d, d))  
    Q = (Q + Q.T) / 2  
    Q = Q - np.mean(Q)
```

```
def blackbox(x: np.ndarray) -> float:  
    assert x.shape == (d,) # x は要素数 d の一次元配列  
    return x @ Q @ x # type: ignore
```

```
return blackbox
```



```
def make_blackbox_func(d: int) -> Callable[[np.ndarray], float]:  
    # 必要に応じて最適化対象以外の条件設定や初期設定などをここで実施
```

```
def blackbox(x: np.ndarray) -> float:  
    # x の組合せを元の実験を実施  
    # または、シミュレーションを実施  
    # 実験・シミュレーション結果を後処理し、最小化したい目的関数値を取得  
    objective = hogehoge  
    return objective # type: ignore
```

```
return blackbox
```

基本的に、`blackbox()`を変更する。必要に応じて、現在の教師データの出力などを追加。

- 例① : `blackbox()` 内でシミュレーションを呼び出し、後処理、その戻り値を最小化するように最適化。
- 例② : `blackbox()` 内で実験を行う。つまり、1回のFMQA で推定された探索候補 \hat{x} を対象に実験し、結果を教師データに追加、次のFMQA 試行を行う。
- $h(x)$ を最大にするような入力 x を推定する場合は、 $1/h(x)$ や $-h(x)$ などを目的関数 $f(x)$ とする。

今後について

ぜひ、デモ・チュートリアルにあるサンプルコードにも挑戦してみてください！

一般的な組合せ最適化問題

目的関数のみで
定式化



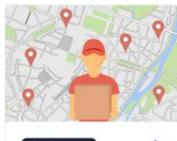
チュートリアル基礎編
画像のノイズ除去
プログラミング難易度 ★★★★★
画像のノイズ除去を行うアプリケーションを開発します。
サンプルコード

制約条件のみで
定式化



チュートリアル応用編
会議室割当問題
プログラミング難易度 ★★★★★
制約条件を用いて定式化するアプリケーションの例として会議室割当問題のアプリケーションを開発します。
サンプルコード

目的関数 + 制約条件



デモアプリケーション
巡回セールスマン問題
プログラミング難易度 ★★★★★
Fixstars Amplifyによる、巡回セールスマン問題の定式化を体験します。
デモアプリ サンプルコード



デモアプリケーション
容量制約つき運搬経路問題 (CVRP)
プログラミング難易度 ★★★★★
複雑で未知な目的関数にも適用可能な、機械学習と組み合わせた最適化を組み合わせた容量制約つき運搬経路問題 (CVRP) を取り扱います。
デモアプリ サンプルコード

ブラックボックス最適化問題

概要



チュートリアル応用編
ブラックボックス最適化 (1)
プログラミング難易度 ★★★★★
複雑で未知な目的関数にも適用可能な、機械学習と組み合わせた最適化を組み合わせたブラックボックス最適化手法を紹介し、Amplifyを用いて実装します。
サンプルコード

材料探索



チュートリアル応用編
ブラックボックス最適化 (2)
プログラミング難易度 ★★★★★
機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超伝導を実現する材料探索を取り扱います。
サンプルコード

翼型最適化



チュートリアル応用編
ブラックボックス最適化 (4)
プログラミング難易度 ★★★★★
流体力学設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と実体シミュレーションを用い、翼の屈曲比を最大化するように翼型の探索を行います。
サンプルコード

信号機制御



チュートリアル応用編
ブラックボックス最適化 (5)
プログラミング難易度 ★★★★★
ブラックボックス最適化により、商業施設による交通渋滞が発生し得る都市における、交通渋滞を低減するような信号機制御の最適化を実装します。最適化の実施及び検証には、マルチエージェント・シミュレーションによる交通シミュレーションを用います。
サンプルコード



困った時はドキュメンテーションを！

<https://amplify.fixstars.com/docs/amplify/v1/index.html>

今後のセミナーの予定

今後も定期的に無料セミナーを開催します！

2025/7/22

ブラックボックス最適化 (材料探索)

- ・はじめに
- ・会社紹介
- ・Fixstars Amplifyの紹介
- ・ブラックボックス最適化のワークショップ
- ・Wrap Up
 - ・事例のご紹介
 - ・今後の進め方
 - ・Q&A

2025/8/21

ブラックボックス最適化 (攪拌機器の最適設計)

- ・はじめに
- ・会社紹介
- ・Fixstars Amplifyの紹介
- ・ブラックボックス最適化のワークショップ
- ・Wrap Up
 - ・事例のご紹介
 - ・今後の進め方
 - ・Q&A

2025/9/25 (2枠実施！)

最新事例と技術解説 (Annealing Engine)

- ・はじめに
- ・Fixstars Amplifyのご紹介
- ・最新事例紹介
- ・技術解説
 - ・制約条件の重み
 - ・求解時間の設定
 - ・多目的最適化
 - ・その他、躓きポイント

ご質問・ご不明点がありましたら、お問い合わせフォームでご連絡下さい

<https://amplify.fixstars.com/ja/contact>

Q&A